AIN SHAMS UNIVERSITY FACULTY OF ENGINEERING

Fall 2023

Name	ID
Omar Medhat Mohamed Mousa	2100271
Abdelrahaman Reda Abdelrahman	2100870
Haya Mahmoud Abdelhamid	2100548
Ahmed Saad Mohamed	2100266
Mohamed Nasser Mohamed	2201091
Yousif Jaber Mohamed	1901037

Source Code:

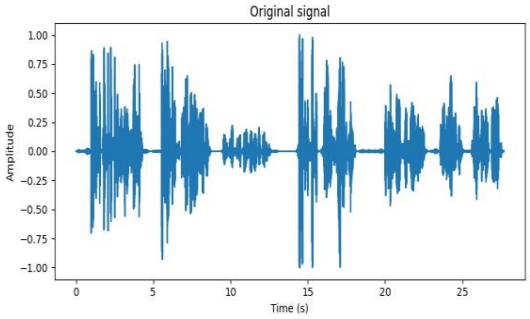
```
import PySimpleGUI as sg
import scipy.signal as signal
import soundfile  as  sf
import numpy as np
import matplotlib.pyplot as plt

# Define the GUI layout
layout = [
    [sg.Text("Select an audio file:")],
    [sg.Input(key="-FILE-"), sg.FileBrowse()],
    [sg.Text("Select a filter type:")],
    [sg.Radio("Low-pass", "FILTER", key="-LPF-"), sg.Radio("High-pass",
"FILTER", key="-HPF-")],
    [sg.Text("Enter the cutoff frequency (in Hz):")],
    [sg.Input(key="-FREQ-")],
    [sg.Button("Filter"), sg.Button("Exit")]
]

# Create the GUI window
window = sg.Window("Audio Filter", layout)

# Event loop
while True:
    event, values = window.read()
    if event == "Exit" or event == sg.WIN_CLOSED:
        break
    if event == "Filter":
        # Get the user input
        file = values["-FILE-"]
        lpf = values["-LPF-"]
        hpf = values["-HPF-"]
        freq = float(values["-FREQ-"])

        # Read the audio file
        data, samplerate = sf.read(file)
        # Convert to mono if stereo
        if data.ndim == 2:
            data = data.mean(axis=1)
        # Get the time axis
        time = np.arange(len(data)) / samplerate

        # Plot the original signal in time domain
        plt.figure()
        plt.subplot(2, 2, 1)
        plt.plot(time, data)
        plt.xlabel("Time (s)")
        plt.ylabel("Amplitude")
        plt.title("Original signal")

        # Compute the FFT of the original signal
        freqs = np.fft.rfftfreq(len(data), 1/samplerate)
        fft = np.fft.rfft(data)
```

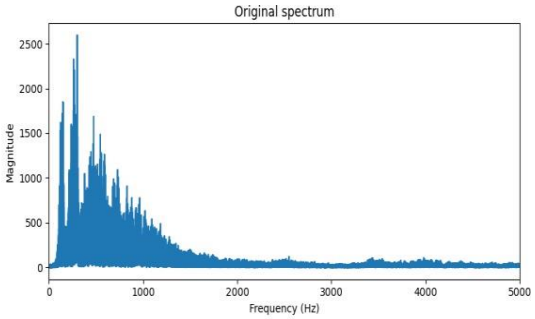
```
        # Plot the original signal in frequency domain
        plt.subplot(2, 2, 2)
        plt.plot(freqs, np.abs(fft))
        plt.xlabel("Frequency (Hz)")
        plt.ylabel("Magnitude")
        plt.title("Original spectrum")

        # Design the filter
        if lpf:
            b, a = signal.butter(4, freq, "low", fs=samplerate)
        elif hpf:
            b, a = signal.butter(4, freq, "high", fs=samplerate)
        else:
            sg.popup("Please select a filter type")
            continue

        # Apply the filter to the original signal
        filtered_data = signal.filtfilt(b, a, data)

        # Plot the filtered signal in time domain
        plt.subplot(2, 2, 3)
        plt.plot(time, filtered_data)
        plt.xlabel("Time (s)")
        plt.ylabel("Amplitude")
        plt.title("Filtered signal")

        # Compute the FFT of the filtered signal
        filtered_fft = np.fft.rfft(filtered_data)

        # Plot the filtered signal in frequency domain
        plt.subplot(2, 2, 4)
        plt.plot(freqs, np.abs(filtered_fft))
        plt.xlabel("Frequency (Hz)")
        plt.ylabel("Magnitude")
        plt.title("Filtered spectrum")

        # Show the plots
        plt.tight_layout()
        plt.show()

        # Save the filtered signal as a new audio file
        new_file = file[:-4] + "_filtered.wav"
        sf.write(new_file, filtered_data, samplerate)
        sg.popup(f"Filtered audio file saved as {new_file}")

# Close the GUI window
window.close()
```

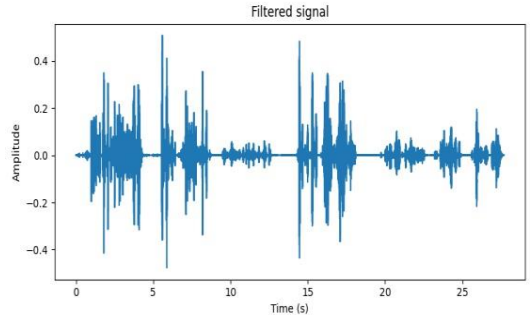
ScreenShots

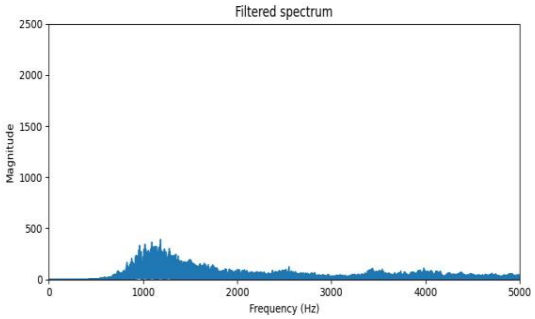
1.HPF

Cutoff F = 1000Hz.









2.LPF

Cutoff f = 1500 Hz